Krill: An Architecture for Edge-Heavy Data

Abstract

We argue that most of data will be stored and processed at the edge of the network in the next "BigData" era. We call this phenomenon "Edge-Heavy Data", and we propose an architecture named Krill[1] *for it. One important characteristic of BigData is its low value-density. If the data are never used, it is wasteful of sending it to the data centers (with a network cost) and storing them into expensive enterprise-grade data servers. Instead, the low value-density data will be stored near where they are generated, that is, the edge of the network.*

This paper discusses requirements and a possible architecture for efficiently dealing with the phenomenon of edge-heavy Data. We first consider a few scenarios where edge-heavy data is desirable, or even inevitable, and identify its requirements. Then we propose an architecture based on the concept of Data Value Field (DVF), followed by an introduction of Jubatus, an open-source framework of online machine learning, as a first step towards the proposed architecture.

1. Introduction: Motivating Scenarios

We argue that most of data will be stored and processed at the edge of the network in the next "BigData" era. We call this phenomenon "Edge-Heavy Data", and we propose an architecture for it.

Why most of the data will be stored and processed at the edge of the network? We believe that there are at least three drivers: cost, privacy, and latency. In the following subsections we consider three scenarios, (1) surveillance cameras, (2) bio-sensors, and (3) personal mobility that motivate these requirements.

1.1. Surveillance Cameras

Surveillance cameras are our first example. The camera data are recorded and stored to locally-installed digital recorders. A typical recorder has 0.5-4TB storage capacities and can keep the recorded video for between a few days and a few months. The data are rarely sent to a central data center because of its volume and the cost of transmitting it to the data center and storing it in the storage. The worldwide market of surveillance cameras in 2011 is said to be 7.5M units. If we assume that each camera is allocated 100GB of storage, the total size is 750PB, which is already too large to collect or store into one place.

Each camera has its own "value-density," that is, the value per bit. The data of the front entrance where many people pass by are much more valuable than the data from a camera that monitors the activities of the backyard parking lot where the activities are infrequent and far in between. The value-density can also vary depending on the time of the day, or the day of the week. It is desirable that the data will be distributed among multiple recorders so that the overall storage capacity can be maximally utilized depending on the value-density of the data.

The data of a camera should be associated with those of the cameras nearby. For example, if we want to track a movement of a particular person (e.g., a potential terrorist), this information is summarized and distributed to neighborhood cameras where this person has a high probability of showing up. This enables efficient tracking of the movement of the person.

Note that the same tracking is possible if all the camera data are available on a centralized server. The cost of sending and storing the data is, however, prohibitively large, especially considering that these events of interests happen relatively rarely. Thus, the network and storage cost will be the major driver for edge-heavy data in this scenario.

1.2. Vital Sign Monitoring

Monitoring of vital signs, such as blood pressure, pulse, body temperature, and respiration sensors is becoming less expensive and ubiquitous. The collected data can be used for monitoring and improving the health individually, but the data can also be shared within a group of people to do statistical analysis and predict and control diseases for the person and the group.

One of the biggest obstacles for doing so is privacy. If the data are sent to a central server and stored there, the risk of compromising is considered to be high. It is desirable that the raw data stay in the device owned by the individual as much as possible, and only statistical information is shared with limited neighbors in the group. Thus, privacy-sensitive data need to be edge-heavy. The individual can obtain other's statistics according to their privileges. For example, people can know their family's status, and epidemic researchers can access the statistical information of people in a certain local area.

This statistical information sharing may not be symmetric. For example, parents may want to know the rate of flu infection in the school of their children, but not vice versa.

1.3. Personal Mobility

It is predicted that a large number of personal mobility vehicle (e.g., electric wheelchairs, Segway, or more advance prototypes such as Honda 3R-C) will be used in the near future.

[1] Antarctic Krill, with its $10^{14}+$ population, is considered to have the largest animal biomass on earth.

These units collect huge amount of data from their sensors and interact with other units to share real-time information for improving the safety, efficiency, and overall utility for their passengers.

The key requirement in this scenario is low-latency. You need the location and velocity information of neighboring units that may appear in the next intersection before you get there in order to avoid collision. Because of the network latency and the potentially large number of such units operating at the same time, it is unrealistic to send all the data to centralized servers and redistribute them back to the pertinent units. Instead, the data should be stored and shared locally with nearby units.

1.4. Other Scenarios

In 2012, more than 722 million smartphones were shipped worldwide according to IDC's report . Smartphones have a large storage capacity. If we assume that 10GB is available to store sensor data in a smartphone, the overall storage capacity at the network edge in the world has increased to 7.2 exa bytes in 2012. Compared to 6.7 exa bytes, estimated amout of new data stored worldwide in 2010 , more than half of new data could be readily stored in the network edge.

In astronomy, large amount of data are continuously generated from various telescopes worldwide. The data are stored in data centers of each organization, but they can not be globally consolidated into a centralized server due to its sheer size. Instead, the data are kept in the distributed data centers. This is another form of edge-heaviness.

2. Krill : Architecture for Edge-Heavy Data

2.1. Data Value Field (DVF)

The essence of the edge-heavy data phenomenon is that the value of data is dynamically determined by the stakeholders, and the data will be optimally allocated over the network based on the values. The way to represent the value of data becomes the key question. The value of data may consist of its present value as well as its future (speculative) values. The data from a surveillance camera may not be interesting at present, but if later a suspect is found to be in its vicinity, the data value suddenly increases.

We define Data Value Field (DVF) as a tuple $< A, V >$ where $A = \{a_1, a_2, \ldots, a_N\}$ are N agents called *krills* and V is its *predictive value model*. Each krill a_i has a time series data collected from its locally attached sensors. We define the predictive value model $v_{t_0}(i, j)$ as representing the value of data a_j looked at by the krill a_i. In general, the data values are asymmetric, that is, $v_{t_0}(i, j) \neq v_{t_0}(j, i)$. Figure 1 shows a DVF consisting of two krills a_1 and a_2.

The predictive value model $v_{t_0}(i, j)$ where the data is taken at time t_0 is defined as a cone of uncertainty; the model gives a probabilistic distribution of the value of the data item of krill a_j collected at time t_0, and this distribution has larger variation (uncertainty) as t increases as shown in Figure 2.

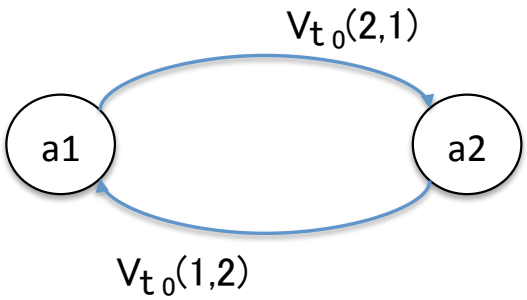

Figure 1: Data Value Field

The present value of the data item collected at time t_0 is the sum of the values of all the krills, including the speculative values for the future values at time t. Each krill determines the strategy for dealing with the data item at time t_0 based on the value of the data. If the value is estimated to be low, the krill may decide to degrade the data (i.e., decreasing the resolution if it is a video data), or even discard it. If the data value is extremely high, that is, the data is very likely to be used by other krills, the data may be speculatively distributed to nearby krills to optimize the data access.

There are multiple ways to represent a DVF. One is to represent a DVF using a directed graph as shown in Figure 3a. Each node represents a krill (and its associated time-series data) and each edge represents the value of the data at the distination node, looked at from the source node. If the krills are located in an Eucridean space, the values could be defined based on the distance between two krills in the space (Figure 3b). This is adequate for transportation data.

2.2. Probabilistic Programming Model

In our programming model, when we access a data a_j, the data may or may not be available or in low resolution (or with a high variance) because of the low valuation of the data. Or, high-value data may be cached in nearby nodes. In our programming model, we would like such high-value data to be always available locally so that the programmer can be relieved from the details of the data management.

We assume that all the data a_i and any variables derived from them are stochastic, meaning that the variable follows a prabability distribution. For example, in a statement

$$Y = X + 3$$

X and Y are random variables if X is a data item from some sensor.

Each krill has a set of *functions* that define the behavior of the krill. A function $y = f_i(x_1, x_2, ..., x_n)$ computes the probablistic distributions of the output variables y from the input random variables $x_1, x_2, ..., x_n$. These distributions may be parametric (e.g., Gaussian distribution) or non-parametric.

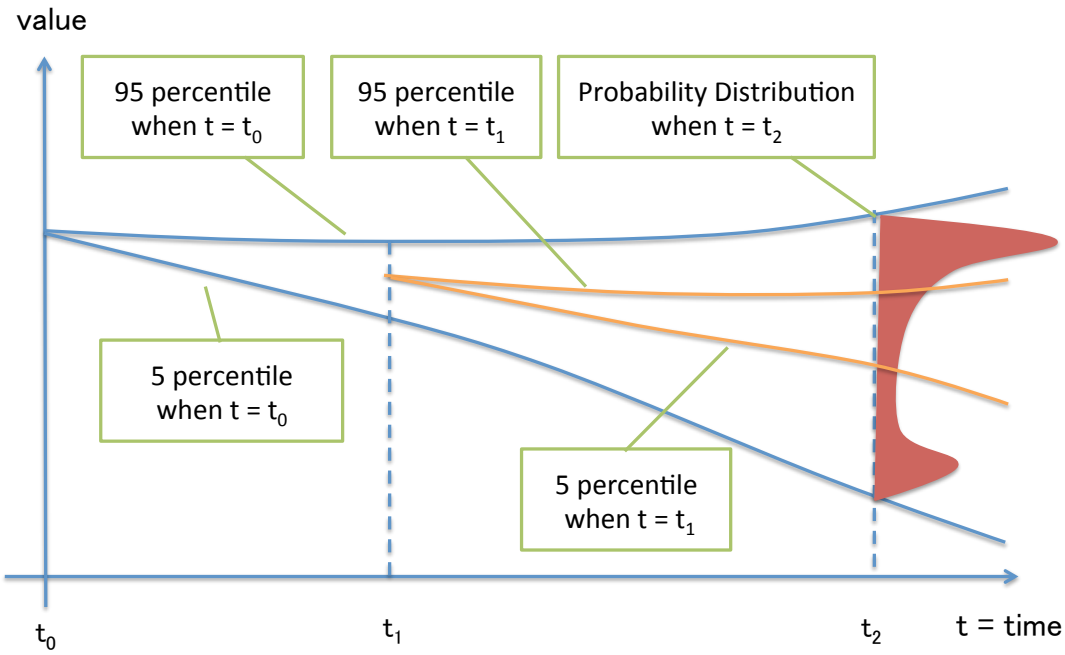

Figure 2: Cone of Uncertainty

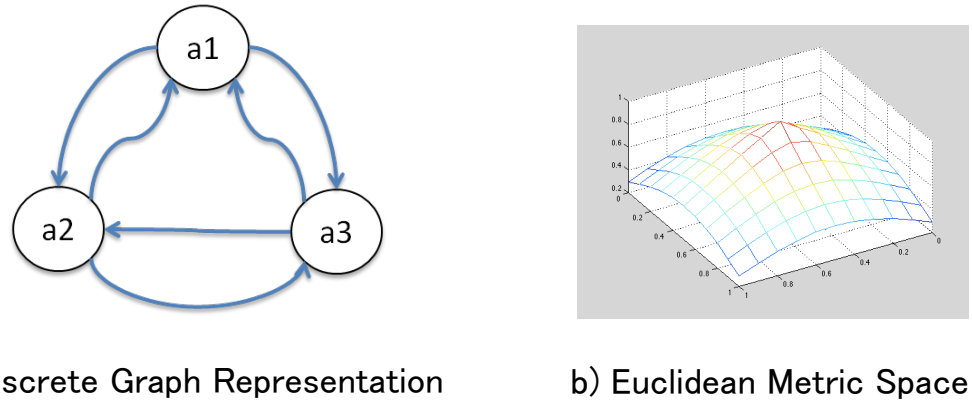

Figure 3: DVF Representations

They are given as analytical functions for known distributions, or can be computed according to existing statistical methods such as Kalman filter or particle filter .

An input variable should have a *prior* distribution in case when the value is not available or the accuracy is very low (a high variance). This enables robust computation even if no data is available.

Each krill is associated with a global ID, such as a URL. This ID can be used to access the sensor data of other krills. If we would like to intensionally specify a set of krills, for example, "all the temperature data of the krills within 100 meters," we use a query language to obtain the extensive list. The availability and the resolution of each data will be determined dynamically at the runtime.

2.3. Runtime System

Each krill implements five sets of tasks as follows:

- Sensor: Collects data from the sensor(s).
- Storage: Manages the local storage of the krill based the predictive values of the data. If the predictive value is low, it can be discarded, or reduced in size by replacing it with a low-resolution (high variance) data. This task optimizes the use of the local storage.
- Distributor: Shares its own data with other krills based on its predictive values, subject to its access control policy. If the data have high value to other krills, the data may be replicated to other krills.
- Aggregator: Requests necessary data to other krills based on its own DVF and program execution. Note that the data

may not be available in a timely fashion. In case the data are not available, the prior distribution is used as the default. Multiple distributors and aggregators collectively optimizes the global utilization of the storage and the network bandwidth.

- Processor: Executes application programs (functions).

Because the data quality (resolution and availability) is transparent to the programmers, there are a number of opportunities to optimize the data representations and data locations globally.

2.4. Device Architecture

The probabilistic nature of the programming model opens an interesting opportunity for device architecture, because now the storage on a device can be less than 100% reliable. Some data have smaller predicted values and can be stored in less-reliable memory. It is known that the current solid-state disks (SSDs) incurrs a large overhead for hiding the low reliability of the flash memory both in terms of hardware cost and power consuption. The relaxed requirement of the storage reliability may enable significant reductions of cost and power.

3. Jubatus: a step towards Edge-Heavy Data

We introduce Jubatus as the first step towards realizing the programming model mentioned above.

Jubatus categorizes its data into two classes based on ther value-density: raw data and models. While the value-density of raw data is low and cannot be shared among krills, that of model is high, and can be shared. Jubatus concentrates value of data by analyzing and summerizing it locally, and enable us to distribute important information without transmitting low-valued data.

Jubatus is an open-source software framework for online machine learning in distributed environments. Machine learning corresponds to a set of algorithms for understanding and predicting data characteristics by extracting hidden rules inside BigData using statistical methodologies. The central idea of Jubatus lies in the separation of raw data from the model learnt from them, which consists of statistical parameters that represent the summarized information of the data.

By using the distributed and online learning algorithms in Jubatus, every krill builds a learning model and trains it locally, and shares it with others to obtain the best possible model. Online learning capability enables us to take data samples one-by-one, iteratively update the model, and keep the model up-to-date. The models will be shared between the krills, integrated into a unified model, and sent back to all of the krills.

The main advantage of Jubatus beyond other learning platforms lies in the separation of shared data from raw data. It simply gives up sharing raw data samples used to train each local model, but instead the trained models are exchanged with other krills. Though the types and structures of machine learning models vary depending on the tasks and algorithms, most of them can be represented as a fixed set of learned parameters on specific type of model. Therefore, the memory size for storing such model is far smaller than that for storing all of the data samples in main memory so that the model information can be quickly transmitted between the krills with much smaller networking cost. At a cost of implementing each learning algorithm with special care for sharing and integrating the model, this allows us to improve the efficiency of handling edge-heavy data by the cooperation of distributed krills.

According to these techniques, Jubatus is very good at real-time analytics, which is fundamental in many edge-heavy data applications. It is possible to update the model and use it for prediction in real-time since such processes can perform sequentially and locally on each krill, and Jubatus does not need any parallel or distributed computations during updating and using the model. Adding more krills simply improves the throughput performance of the whole learning system.

As an abstraction of the computational model, Jubatus depends on three phase processes, UPDATE-MIX-ANALYZE, for any machine learning tasks such as classification, regression, recommendation, or anomaly detection. The UPDATE defines how to update the model, and the ANALYZE defines how to analyze the data using the local model, and the MIX defines how to mix the models in different krills. The key to tackle edge-heavy data problems is to develop efficient and effective logics for MIX operations. They will allow Jubatus to run on any kinds of edge devices, speculatively and selectively collect data that are valuable to themselves at the time from their neighborhoods, and become intelligent enough to make correct decisions in the current local situation. These techniques will be the future of Jubatus in edge-heavy data era.

4. Future Directions

The idea of Edge-Heavy Data arose when we were discussing the future architectures for BigData. We believe that edge-heaviness is an inevitable consequence to deal with ever-increasing BigData in the wild. There are a number of challenges to realize this new architecture in working systems. Let us mention two fundamentally important items of them.

First, we need to define the details of the programing model. A very few attempts have been made so far to make programming inherently stochastic as to the authors' knowledge. Sound and complete definition of the language as well as techniques for efficient implementation must be developed.

Second, the runtime system requires a lot of optimization of data allocation by taking into account various aspects of the system, including network parameters, node processing parameters (computation power, storage capacity, power consumption, etc.), and application workload. Jubatus is a first step towards this optimization.

References

[1] A. Doucet and N. de Freitas, *Sequential Monte Carlo methods in practice*. Springer, 2001.

[2] IDC, "Worldwide quarterly mobile phone tracker," Tech. Rep., 2012.

[3] Jubatus Development Team, "Jubatus: Distributed online machine learning framework," http://jubat.us/en/.

[4] McKinsey Global Institute, "Big data: The next frontier for innovation, competition, and productivity," Tech. Rep., 2011.